Mathematics 4MB3/6MB3 Mathematical Biology

http://ms.mcmaster.ca/earn/4MB3

Fall 2024 ASSIGNMENT 4

This assignment is due on Crowdmark on **Monday 11 November 2024 at 11:59pm**. Do not submit a hardcopy. In addition, a compressed archive of all your source files must be e-mailed to the instructor and TA (unless you choose to invite the instructor and TA to an associated `github` repository).

General Notes

(i) **Please re-read all the General Notes listed for previous assignments.**

(ii) **Please re-read all the Technical Notes listed for previous assignments.**

1 Time Series analysis of Recurrent Epidemics

(a) You should have received the following data files by e-mail:

`meas_uk__lon_1944-94_wk.csv`
`meas_uk__lpl_1944-94_wk.csv`

These plain text comma-separated-value files list weekly cases of measles (in London and Liverpool, England, from 1944 to 1994). Depending on which research direction you select, you might receive other files in the same `ymdc` (year,month,day,count) format, where the count column might contain cases or deaths, for example. Write the following R functions:

(i) `read.ymdc()`. Read a file in `ymdc` format and return a data frame containing these data and including a `date` column that has R's `Date` class. The first (and potentially only) argument to this function should be the `filename` of the data file to be read.

(ii) `time.plot()`. Given a data frame produced using `read.ymdc()`, display the associated time plot. The first argument of the function should be the data frame. Further optional argument(s) should allow the user to smooth the time series with a moving average. By default, this function should create a new plot but there should be an option to add to an existing plot. Implement this by having a logical `add` argument that is false by default (`add=FALSE`). This will allow you to add a smoothed version of the time series on top of the raw data, for example. The final argument should be the ellipsis (`...`) so that details such as colour and line style can be passed to the plotting commands used in this function.

(iii) `periodogram()`. Given a data frame produced using `read.ymdc()`, display the associated *period periodogram* (power spectrum as a function of period). The first argument of the function should be the data frame. By default, the entire time series should be used, but optional argument(s) should allow the user to specify a time range of interest. Use R's `spectrum()` function to compute the power spectrum. Have `add` and `...` arguments as in `time.plot()`. Note that if `v` is a vector containing a time series of interest, you can obtain and plot its *frequency* periodogram as follows.

```
s <- spectrum(v, plot=FALSE)
plot( s$freq, s$spec, type="l")
```

(b) Using your functions, make a multi-panel plot that clearly shows the temporal pattern of the time series and how its frequency structure changes over time. Think carefully about how to make this multi-panel figure as clear as possible for yourselves and your readers. Describe your figure, explaining what aspects of your figure you feel are puzzling or interesting and may be possible to understand using mechanistic mathematical modelling. (Repeat this for each of the epidemic time series you are given.)

2 Stochastic Epidemic Simulations

Consider the SI model,

$$\frac{dI}{dt} = \beta(N - I)I\,, \qquad I(0) = I_0, \tag{1}$$

where β is the transmission rate, N is the population size and $I(t)$ is the number of infected individuals at time t.

(a) Write an R function `SI.Gillespie()` that uses the Gillespie algorithm to produce a realization of a stochastic process whose mean field dynamics are given by equation (1) in the limit $N \to \infty$. Your function should have arguments `beta`, `N`, `I0` and `tmax` (the time at which to end the simulation). You may find it helpful (conceptually) to write equation (1) in two-variable form:

$$\frac{dS}{dt} = -\beta SI\,, \qquad S(0) = N - I_0, \tag{2a}$$

$$\frac{dI}{dt} = \beta SI\,, \qquad I(0) = I_0. \tag{2b}$$

Note that there is only one type of event that can occur, so the second part of the Gillespie algorithm (what type of event occurred) is trivial for this model.

(b) Make a multi-panel plot comparing the deterministic and stochastic dynamics of the SI model for $\beta = 1$, $I_0 = 1$ and $N \in \{32, 10^2, 10^3, 10^4\}$ ($N = 32$ is close to $10^{1.5}$). Each panel should correspond to a different value of N and should show 30 stochastic realizations together with the deterministic solution.

Note: To make stochastic simulations exactly reproducible use `set.seed()`.

3 $\mathcal{R}_0$ for smallpox

The natural history of smallpox is shown in Figure 1. The US Centers for Disease Control and Prevention (CDC) has recently discovered that a group of bioterrorists plans to reintroduce smallpox to the United States. The CDC has reason to believe that the terrorists are also bioengineers and have successfully altered the virus so that it causes the early rash stage to be twice

as long as it was when the virus was last circulating naturally in the 1970s. Moreover, the existing smallpox vaccine apparently provides no protection against the altered virus. The CDC wants your opinion on how the alterations to the virus will affect $\mathcal{R}_0$ and the expected final size of an epidemic if the planned attack is successful.

(a) Construct a compartmental (ODE) smallpox transmission model based on the natural history specified in Figure 1, including vital dynamics but ignoring disease-induced death.

(b) Use a biological argument to find a formula for $\mathcal{R}_0$.

(c) Calculate $\mathcal{R}_0$ using the next generation matrix approach. *Note:* Your solution should include $\mathcal{F}$, $\mathcal{V}$, F, V, and FV^{-1}, in the most human-friendly form you can find. However, feel free to use symbolic manipulation software such as `Maple`, *Mathematica* or sage to help with the necessary algebra and matrix computations.

(d) Based on your model, and $\mathcal{R}_0 \sim 5$ for unaltered smallpox, what can you say about the difference in $\mathcal{R}_0$ that can be expected for the newly engineered virus vs. the original virus?

(e) Write a paragraph that you can imagine e-mailing to the CDC, in which you do your best to answer their questions.

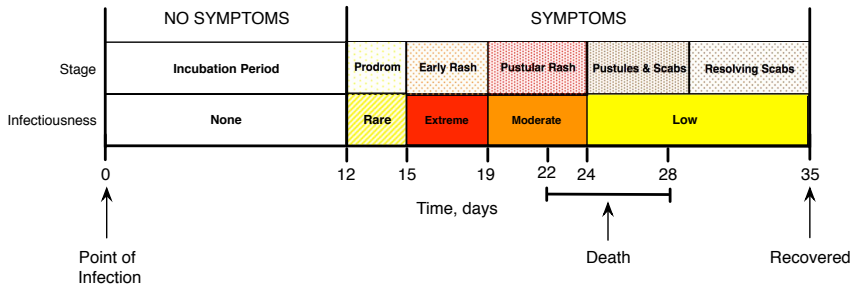

Figure 1: The natural history of smallpox infection. The prodrom stage begins with fever but the patient is very rarely contagious. Early rash is the most contagious stage, when the rash develops and transforms into bumps. During the pustular rash stage bumps become pustules, which then turn into scabs during the pustules and scabs stage and fall off during the resolving scabs stage. The infected person is contagious until the last scab falls off. (*This is Figure 3.4 from page 82 of Olga Krylova's 2011 McMaster University PhD thesis.*)

4 `epigrowthfit` (BONUS PROBLEM)

This problem leads you step by step through a simple application of Mikael Jagan's `epigrowthfit` package,

https://cran.r-project.org/web/packages/epigrowthfit/,

which you learned about in Mikael's guest lecture.

Suppose that the time evolution of expected cumulative incidence during an epidemic follows a generalized logistic (or "Richards") model,

$$c(t;\theta) = \frac{K}{[1 + a\exp(-ar(t - t_{\text{infl}}))]^{1/a}}\,, \quad \theta = (\log r, \log t_{\text{infl}}, \log K, \log a) \in \mathbb{R}^4\,, \tag{3}$$

where $r, t_{\text{infl}}, K, a > 0$ represent an initial exponential growth rate, inflection time, final size, and "shape" parameter, respectively. The standard logistic model is the special case obtained by setting $a = 1$.

(a) Write an R function `richards(t, theta)` to compute $c(t;\theta)$. It must be "vectorized" in `t`, so that `length(richards(t, theta))` is equal to `length(t)`. Note that θ specifies the "natural" parameters r, t_{infl}, K, a on an unconstrained logarithmic scale: the domain is all of $\mathbb{R}^4$ rather than the subset $\mathbb{R}^4_{>0}$. (This is a deliberate choice: optimization over an unconstrained domain tends to be more robust to numerical errors and is supported by a broader class of optimization algorithms.)

(b) Write an R function `richards.sub(s, t, theta)` to compute expected incidence during time intervals $(s, t]$:

$$f(s, t; \theta) = c(t;\theta) - c(s;\theta)\,. \tag{4}$$

It must be vectorized in `s` and `t`. You can assume that `length(s)` and `length(t)` are equal.

(c) Write an R function `richards.nll(s, t, x, theta)` to compute the negative log likelihood $L(\theta)$ of the model θ given some data $\{(s_i, t_i, x_i)\}_{i=1}^n$, where x_i is a count of infections over a time interval $(s_i, t_i]$. Suppose that x_i is realization of a random variable X_i distributed as

$$X_i \sim \text{NegativeBinomial}(\mu = f(s_i, t_i; \theta), k = 20000)\,. \tag{5}$$

In other words, X_i follows a negative binomial distribution with mean $\mu = f(s_i, t_i; \theta)$ and "dispersion" $k = 20000$ (implying variance equal to $\mu(1 + \mu/20000)$). To compute negative binomial probabilities, use function `dnbinom`; see `help("dnbinom")` for usage details.

(d) Use function `rnbinom` to simulate counts $\{X_i\}_{i=1}^n$ distributed as above, taking $s_i = i - 1$, $t_i = i$, $n = 32$, $r = 0.08$, $t_{\text{infl}} = 22.5$, $K = 10000$, and $a = 1$. Evaluate `set.seed(16131)` before calling `rnbinom` so that your simulation is reproducible. To verify that the `rnbinom` result is reasonable, print $\{x_i\}_{i=1}^n$ and plot (in separate panels, as points) the corresponding incidence time series $\{(t_i, x_i)\}_{i=1}^n$ and cumulative incidence time series $\{(t_i, \sum_{j=1}^i x_j)\}_{i=1}^n$. Functions `print`, `plot`, and `cumsum` will be useful here.

(e) Compute $L(\theta)$ for $\theta = \theta_0$ defined by $r = 0.08$, $t_{\text{infl}} = 22.5$, $K = 10000$, and $a = 1$ as well as for $\theta = \theta_1$ defined by $r = 0.1$, $t_{\text{infl}} = 20$, $K = 12500$, and $a = 1$.

(f) Write an R function `richards.nll.opt(s, t, x, thetaInit)` returning a list containing a local minimum point $\hat{\theta}$ of L given data $\{(s_i, t_i, x_i)\}_{i=1}^n$ supplied by `s`, `t`, and `x`. Use function `optim` to obtain such a list, initializing the optimizer with `thetaInit`, the user's initial guess of $\hat{\theta}$. Hence `richards.nll.opt` should return the value of `optim(par = thetaInit, fn = <some function>)`, and the main challenge for you is to define `fn` appropriately. For guidance, do consult `help("optim")` and the code examples from Mikael's lecture on October 1.

(g) Use `richards.nll.opt` to compute $\hat{\theta}_0$ and $\hat{L}_0 = L(\hat{\theta})$ given the simulated data, starting from θ_1. We are using subscript 0 here to emphasize that $\hat{\theta}_0$ estimates θ_0, the data-generating parameter vector. $\hat{\theta}_0$ and $\hat{L}_0$ are components `par` and `value` of the list result of `richards.nll.opt`, which you can access using `[[` (as in `object[["par"]]`) or using `$` (as in `object$value`). Note that, due to the possibility of multiple local minima, the error $\theta_0 - \hat{\theta}_0$ may be greater in magnitude than you would expect.[1]

[1] Here, multiple local minima arise due to *nonidentifiability* of r and a. In practice, one would want to penalize $|\log(a)|$ so that optimization favours values of a nearer to 1, i.e., generalized logistic models nearer to the standard one. Doing so, one tends to obtain more accurate estimates of r.

(h) Plot again (as points) your simulated incidence time series $\{(t_i, x_i)\}_{i=1}^n$. Use function `lines` to add (in the same panel, as curves) expected incidence $\{(t_i, f(s_i, t_i; \theta_0)\}_{i=1}^n$ and estimated expected incidence $\{(t_i, f(s_i, t_i; \hat{\theta}_0)\}_{i=1}^n$. Style the curves differently so that they can be distinguished visually. You can do this by adjusting the line type using `lines` argument `lty`. See `help("lines")`. (It directs you to `help("par")` for details about `lty` and other graphical parameters.)

(i) In practice, one must test and not assume that the result of a numerical optimization corresponds to a local optimum point. Discuss how you could test whether $\hat{\theta}$ is a local minimum point of L (or a good approximation of one).

(j) In practice, whether a numerical optimizer converges to a biologically meaningful local optimum point can depend heavily on reasonable starting values for parameters. An estimate of r, the initial exponential growth rate, is the slope of a linear model fit to the logarithm of cumulative incidence—that is, to the points $\{(t_i, \log(\sum_{j=1}^i x_j))\}_{i=1}^m$ for some $m \leq n$ (as growth is exponential only *initially*). Discuss how you could use the cumulative incidence time series $\{(t_i, \sum_{j=1}^i x_j)\}_{i=1}^n$ to obtain an estimate of t_{infl}, the time of inflection in $c(t; \theta)$. Hint: The inflection occurs when the sign of the second derivative $c''(t; \theta)$ changes from positive to negative.

(k) **epigrowthfit** is an R package designed to automate fitting of simple nonlinear models, including the Richards model (3), to incidence time series.[2] Here, you will use **epigrowthfit** function `egf` to reproduce the value of $\hat{\theta}$ that you obtained using your home-grown function `richards.nll.opt`. `help("egf")` is a good reference, but it covers much more general usage, so essential details are repeated here.

Start by installing and attaching the package:

```
install.packages("epigrowthfit")
library(epigrowthfit)
```

[2] Actually, **epigrowthfit** is designed to fit a much broader class of models—nonlinear *mixed effects* models—to *collections* of incidence time series, including time series that span multiple epidemics. Here, we are exploring just the simplest usage: for one time series spanning one epidemic.

Put your simulated incidence time series into a data frame in the format demanded by `egf`:

```
data_ts <- data.frame(time = 0:n, x = c(NA, <your rnbinom result>))
```

The constraints are that `time` must be increasing and that `x[i]` must give the count of infections observed over the interval from `time[i-1]` to `time[i]`. The first element of `x` is always ignored.

Now define a second data frame indicating the "window" of time that you want to consider. Here, we want the entire simulated time series, but, in practice, it is common to have a much longer time series containing segments that we would want to exclude.

```
data_windows <- data.frame(start = 0, end = n)
```

Now define formulae that will tell `egf` how to interpret your data frames.

```
formula_ts <- cbind(time, x) ~ 1
formula_windows <- cbind(start, end) ~ 1
```

The right hand side of `~` is normally used for grouping variables when we deal with collections of time series. Here, `1` indicates "no grouping" and that we are dealing with exactly one time series.

It remains to specify the model that we want to estimate and initial values for the parameters of that model. For a single time series spanning a single epidemic, it turns out that the default initial values are sensible, but, because we aim to reproduce the earlier result, we will override them with θ_1. A complication is that we sometimes want to estimate the negative binomial dispersion parameter k, but here we fix $k = 20000$. This detail must be specified, too:

```
model <- egf_model(curve = "richards", family = "nbinom")
egf_top(model)
[1] "log(r)"     "log(tinfl)" "log(K)"     "log(a)"     "log(disp)"
init <- list(beta = c(log(0.1), log(20), log(12500), log(1), log(20000)))
map <- list(beta = 5) # fix the parameter at index 5
```

We used `egf_top` to determine the relation between elements of `init`[3] and the "natural" parameters of the model (r, t_{infl}, K, a, and the negative binomial dispersion parameter).

We are finally equipped to call `egf`:

```
object <- egf(model = model,
              formula_ts = formula_ts,
              formula_windows = formula_windows,
              data_ts = data_ts,
              data_windows = data_windows,
              init = init,
              map = map)
```

The result is an object of *class* `"egf"` with *methods* for interrogating the estimated model. Use functions `print`, `coef`, and `plot` to print details about the object, extract $\hat{\theta}_0$, and plot the incidence time series contained in `data_ts` alongside estimated expected incidence. These are *generic* functions designed to dispatch methods for the class of the first argument. Documentation for the methods is available as, e.g., `help("plot.egf")`. When calling `plot`, you will want to pass `time_as = "numeric"` and `ylim = c(ymin, ymax)` in order to produce more sensible axes. (Substitute suitable values for `ymin` and `ymax`.)

— END OF ASSIGNMENT —

Compile time for this document: November 4, 2024 @ 16:59

[3] More precisely, component `beta` of the list `init`, where the name `beta` is just a convention.